EXPERIMENT FOUR

Best First Search Algorithm

🔹 Problem Statement:

To find the shortest path from the starting node to a goal node using **Best First Search algorithm** with Python.

```
# Define the graph and heuristic
graph = {
    'A': [('B', 1), ('C', 1), ('D', 1)],
    'B': [('E', 1)],
    'C': [],
    'D': [],
    'E': [('Z', 1)],
    'Z': []
}

# Heuristic function values
heuristic = {
    'A': 21,
    'B': 14,
    'C': 18,
    'D': 18,
    'E': 5,
    'Z': 0
}

# Best First Search Algorithm
from queue import PriorityQueue

def best_first_search(start, goal):
    visited = set()
    pq = PriorityQueue()
    pq.put((heuristic[start], start))
    parent = {start: None}

    while not pq.empty():
        _, current = pq.get()
        print(f"Exploring Node: {current}")
        visited.add(current)

        if current == goal:
            print("Goal reached!")
            break

        for neighbor, _ in graph.get(current, []):
            if neighbor not in visited:
                parent[neighbor] = current
                pq.put((heuristic[neighbor], neighbor))
                visited.add(neighbor)

    # Construct the path
    path = []
    node = goal
    while node is not None:
        path.append(node)
        node = parent.get(node)
    path.reverse()

    return path

# Run the search
start_node = 'A'
goal_node = 'Z'
path = best_first_search(start_node, goal_node)

# Output result
print("\nShortest path using Best First Search:")
print(" → ".join(path))
```

```
Exploring Node: A
Exploring Node: B
Exploring Node: E
Exploring Node: Z
```

```
Goal reached!

Shortest path using Best First Search:
A → B → E → Z
```